Topic No. 9

Epidemic Algorithms

Tim Daniel Hollerung
tdholl(at)uni-paderborn.de
6108355

Peter Bleckmann
blepe(at)uni-paderborn.de
6105561

August, 4th, 2004

Organizer: Christian Schindelhauer

Abstract

Distributing information within networks can be complicated if hosts have only limited knowledge of the properties of the network. This leads to problems when it is highly important that a specific information has to reach one particular host or all hosts within the entire network. Epidemic algorithms follow the paradigm of nature by applying simple rules to spread information by just having a local view of the environment. According to this fact, these algorithms are easy to implement and guarantee message propagation in heterogeneous and not all the time coherent environments.

1 Introduction

In many sections of developing modern algorithms, nature is a paradigm of how fast and efficient algorithms could be designed. A popular example for this is the field of epidemic algorithms, mainly motivated be the need of publishing information in heterogeneous environments, that does not be coherent all the time.

In nature, simple rules define how a specific property or information is distributed. These rules just define methods of how a property or information is exchanged between two participants in the relevant environment. For example take a look at a simple infective disease. The rule how to distribute this disease is, that an infected person *Eve* needs to get in direct contact to an uninfected person *Bob* to infect him with a probability of 85 %.

One of the most popular epidemic algorithms is the *Game of life* invented by John Conway. The environment for this algorithm is a field of cells, where each of them has 8 neighbors as shown in figure 1. The information or property that has to be distributed is the seed of live, meaning that each cell might get occupied based on the following simple rules:

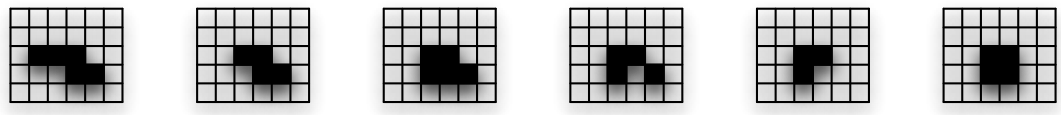

Figure 1: Example for the Game of Life

- Birth: If an unoccupied cell has 3 occupied neighbors, it becomes occupied.
- Death: If an occupied cell has 0 or 1 neighbors, it dies of loneliness, and if it has 4 to 8 occupied neighbors, it dies of overcrowding.
- Survival: If an occupied cell has 2 or 3 neighbors it will survive to the next generation.

In the following sections we will first introduce some models of epidemic algorithms and then introduce some relevant applications of them. Furthermore we present results of their quality and efficiency regarding to their specific application.

2 Classification of epidemic algorithms

In each epidemic algorithm there exists a so-called population, a set of interactive, communicating units. These units use a ruleset that defines how to spread a specific information that might be of interest to other units.

This ruleset if absolutely affected by the design of the algorithm and can be freely chosen. The only requirement is, that at a specific time t a unit must have one of the following states regarding to a specific information:

- Susceptible: the unit does not know anything about the specific information but it can get this specific information
- Infective: the unit knows the specific information and uses the ruleset to spreat the information
- Removed (Recovered): the unit knows the specific information but does not spread it

Based on these definitions, we can define different classes of epidemic algorithms as described in [Sch02]. These classes define how units can handle incoming information in general.

2.1 Susceptible-Infective (SI)

This class defines epidemic algorithms where nearly each unit is initially susceptible. By receiving updated information a unit becomes infective and remains it, until the whole population is infective. Therefore we need some additional methods to decide wether to stop spreading information or not.

Deterministic and continuous model

Let β be the number of contacts a unit has in each round. Assuming, that the number of infective and the number of susceptible units are continuous and not natural numbers, it is possible to show [Sch02], that the relative number of infective units $\#infective(t)$ in round t is

$$\frac{\#infective(0)}{2 \cdot (1 - \#infective(0))} \leq \#infective(t) \leq \frac{\#infective(0)}{1 - \#infective(0)} \cdot e^{\beta \cdot t}$$

until the first half of the population is infective. The number of susceptible units $\#susceptible(t)$ in the same interval in round t is:

$$\frac{1}{2} \cdot \left(\frac{1}{\#infective(0)} - 1 \right) \cdot e^{-\beta \cdot t} \leq \#susceptible(t) \leq \left(\frac{1}{\#infective(0)} - 1 \right) \cdot e^{-\beta \cdot t}$$

In the second half of the population, the number of susceptible units decreases exponentially.

Random directed graphs

For discrete modeling of an SI epidemic the population can be represented by a randomly chosen directed graph $G = (V, E)$, where the set V of nodes represents the population and the set E of directed edges represents the contacts within the considered round. This graph changes every round as each node chooses β new communication partners.

The chance for a particular unit to get infected, is equal to the number of incoming edges of the node mapping to that unit. The possibility of a node to have no incoming edges in one round, meaning not to get infected, is at least $4^{-\beta}$, but smaller or equal to $e^{-\beta}$ for $\beta \leq \frac{n-1}{2}$ as mentioned in [Sch02].

2.2 Susceptible-Infective-Susceptible (SIS)

In difference to the model of SI algorithms, SIS algorithms are able to decide to stop spreading information before the whole population is infective. For example, if a unit realizes that all of his last five communication partners are already infective, it might decide that the specific information, that has to be spread is old, and stops spreading it. But removed units can still become infective again. If a unit gets a specific information, that it has stopped spreading before, it will spread it again until it looses interest again. As shown in [Sch02] and [Kep94], the number of infective units $\#infective$ in round t is:

$$\#infective(t) = \frac{1-p}{1+\left(\frac{(1-p)\cdot n}{\#infectiv(0)}-1\right)} \cdot n$$

where p is the proportion of $\frac{\#removed}{\#infective}$ per round. We can observe, that the number of infective units grows until $\frac{(1-p)}{2}$ units are infective. From that point on, the number of newly infected units decreases exponentially down to $1-p$.

2.3 Susceptible-Infective-Removed (SIR)

SIR algorithms are nearly the same as SIS algorithms with the only difference, that removed units remain removed for a specific information. This means that a unit would never get infective again, after it once stopped spreading a specific information. For instance, a unit, which did not spread a specific information in the last x rounds, stops spreading that specific information for all the time, assuming, that the whole population is already infective. Mathematical modelling of this class is not an easy issue. Up to today, there is no closed formula available for describing the infective rate. There are only numerical analysis available as in [Kep94], that show, as expected, that at the end a constant number of units keep susceptible. This is obvious, as the units stop spreading, because they mainly choose infective units as communication partner and not one of the last remaining susceptible units, that are very few.

3 Enforced ending of rumor spreading

As mentioned in the previous section, all algorithms need at least a method to decide when to stop spreading information. In SIS and SIR algorithms this decision is a more local one, only involving a small part of the population. In simple SI algorithm this decision is not as intuitive, because a unit can not really have a complete overview about what is happing in the whole population.

A good proposal would be to introduce some kind of aging concept. Each specific information will get a counter c_{max} that will be decreased each time the information is transmitted. If this counter is set down to zero, the specific information would not be spread anymore as shown in figure 2. This leads to the natural bound of $c_{max} \cdot n$ messages, where n is the dimension of the population, to spread a specific information.

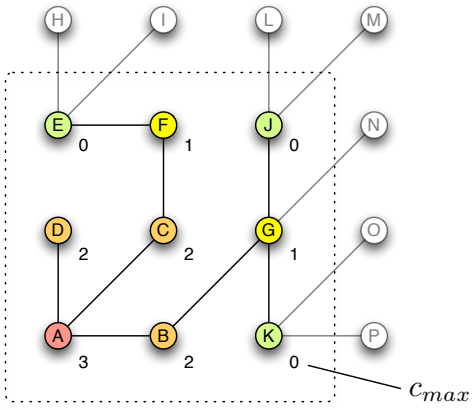

Figure 2: Example for information spreading with $c_{max} = 2$

3.1 Simple methods

The interesting part in the method mentioned above is, how to chose c_{max}. This depends heavily on which method the algorithm uses to spread the information. Especially the direction of the flow of information affects the number of transmission for a specific information to reach the whole population. We will now introduce some communication techniques and show, how long they will take, to infect the whole population.

For the following definitions we will assume the population p with n units using a directed graph G as communication model, that may change every round. Each infective unit of the population chooses uniformly at random one unit of the population (including itself) as a communication partner. This model is close to the discrete modeling of the SI class.

Push algorithms

Each infective unit u sends its new information to its chosen communication partner v (figure 3).

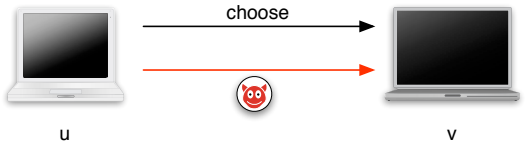

Figure 3: Push explanation

At the beginning, only a few nodes are infective and the chance, to choose a susceptible unit is high. So the chance to infect at least one more units in the first round is more or less reliable. For the first third of the population, the number of infections $\#infective$ in the round t is at least

$$\#infective(t) = 2 \cdot \#infective(t-1) \cdot \left(1 - \frac{c \cdot (\ln n)^2}{n}\right)$$

In the following rounds, when $\#infective(t)$ is in the interval of $[\frac{n}{3}, n]$, the chance for an infective unit to choose a susceptible unit as communication partner gets even smaller. There is not much space left to spread the information. The expected number S(t) of susceptible units in round t is less than:

$$S(t) = e^{-\frac{1}{3} \cdot \#infective(t-1)}$$

As expected these result are very close to the properties of the SI class. In [Pit87] was shown that a push-based SI algorithm needs $\log_2 n + \ln n + o(1)$ rounds to infect the whole population.

Pull algorithms

Each unit u asks its chosen communication partner v, if it has new information. Thus, only if the communication partner v is infective, u would receive new information (figure 4).

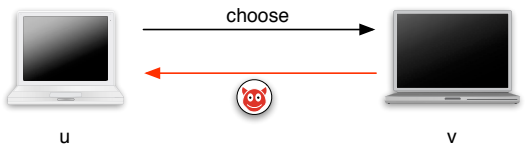

Figure 4: Pull explanation

Due to the fact that only a few nodes are infective at the beginning, the expected relative number $\#infective(t)$ of infective units for round t is:

$$\#infective(t) = 2 \cdot \frac{\#infective(t-1)}{n} - \left(\frac{\#infective(t-1)}{n}\right)^2$$

If #infective(t) is in the interval $[\frac{n}{2}, n]$, the number of susceptible units, that is less or equal to $\frac{1}{2}$, squares for the next $\log\log n + 1$ rounds, up to the whole population is infective:

$$\left(\frac{1}{2}\right)^{2^{\log\log n+1}} < \left(\frac{1}{2}\right)^{logn} = \frac{1}{n}$$

Thus, the expected number of rounds for an pull-based SI algorithm is $\log n + O(\log\log n)$, that is much better than the above results for push-based algorithms.

In comparison to the push communication, the pull communication is at the beginning not very reliable. There is no guarantee, that information spreading begins directly in the first rounds (figure 5).

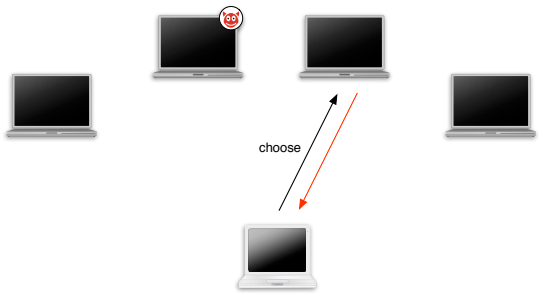

Figure 5: more susceptible units at the beginning

But there is also the chance, that the whole population gets infective during the first round as shown in figure 6. The probability for that is about $\left(\frac{s}{n}\right)^{n-1}$, where s is the number of infective units at the start.

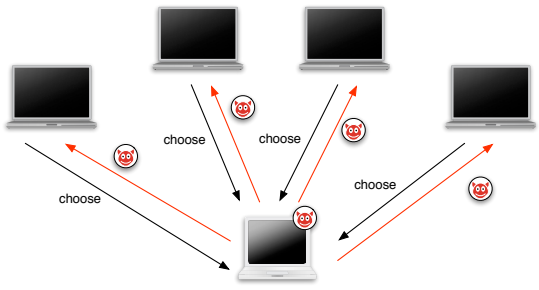

Figure 6: Spreading in one round

At the end, the information spreading using pull communication is very fast and reliable. This is, because the chance to chose an infective communication partner grows round by round.

Push&Pull algorithms

In the beginning of information spreading, push-based algorithm have the advantage, that the chance for an infective unit to choose a susceptible unit as communication partner is clearly higher

than for an susceptible unit to choose a infective one. But if about half of the units are infective, this proportion change in advantage to pull-based algorithms. Now the chance for susceptible nodes to choose an infective one as communication partner grows round by round, while the chance for infective units to choose susceptible ones reduces. In conclusion the pull-based algorithms seems to be more efficient as the push-based algorithms as seen above.

Now a good idea would be to combine the advantages of both techniques, which leads to the model of push&pull algorithms as shown in [KSSV00]. If either the choosing node or the communication parter is infective, both units will be infective (figure 7).

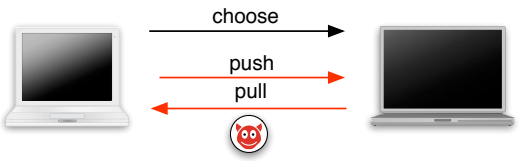

Figure 7: push&pull explanation

If the number of infective units is in the interval of $[1, \frac{n}{2}]$, the push mechanism ensures that after $\log_3 n$ rounds at least half of the population is infective. If the number of infective units is in the interval of $[\frac{n}{2}]$, the pull mechanism ensures, that in $O(\log \log n)$ rounds the second half of the population gets infective.

In conclusion, the infection of the whole population lasts at most $\log_3 n + O(\log \log n)$ rounds. As expected, the push operations dominate the pull operations in the first half of the infection, while in the second half the pull operations dominate, as shown in [KSSV00].

Assuming, that communication partners are not chosen uniformly at random but by a weighted function, push&pull algorithms will even get better results. Within the first rounds, one unit will infect the highest weighted unit via a push operation as the chance, to choose the highest weighted unit as communication partner is very high. As soon as the highest weighted unit is infective, most of all other nodes will get infected via pull operation by choosing this unit as communication partner.

3.2 Shenker's Min-Counter Algorithm

A more efficient technique is the Min-Counter Algorithm introduced by Scott Shenker [KSSV00]. The idea behind the algorithm is, that each unit P gets a counter $c_R(P)$ for each rumor R. This counter is increased if and only if all $c_R(Q_i)$ of the communication partners Q_i from P in the previous round were at least as big as $c_R(P)$. If $c_R(P)$ gets bigger than c_{max}, P would stop spreading rumor R (figure 8).

Analysis show that the whole population can get informed in $\log_3 n + O(\log \log n)$ rounds with a probability of $1 - n^{O(1)}$ using a push&pull algorithm. Therefore only $O(n \log \log n)$ messages need to be send.

4 Replicated Database Maintenance

Databases are a good way to store a huge amount of data and provide it to a user very fast. If there are more than one user at one time, the access speed to the database decreases due to the serialized processing of the queries. In order to get a higher performance on databases that are used by many users, data is replicated and copied to various places near the users.

If now many users want to query for data at the same time, each query can be directed to one of the replicas, to serve the client in time. So, the bottleneck of only having a single database was eliminated resulting in a gain of performance.

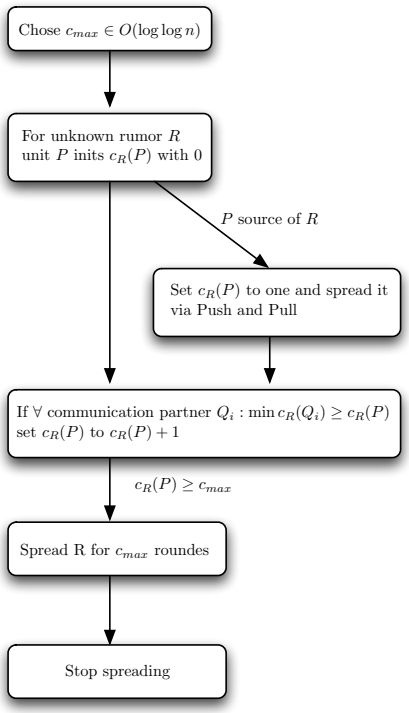

Figure 8: Shenker's Min-Counter Algorithm

In the case, that one or more users write new data to a database problems occur. When one or more of the replicas is changed they will leave the network of databases inconsistent.

4.1 Direct Mail

A straight forward solution to the problem above is the so-called *direct mail* communication: each new update will be send to all other databases by the database in which the update originally took place. This technique is not very reliable, especially not in networks that are not coherent all the time. The database can only update other databases it knows at the time of the update.

To avoid these problems, in [DGH+87] two epidemic algorithms were introduced to extend or replace the direct mail strategy: the anti-entropy and the rumor mongering. In the following sections we will focus on that algorithms.

4.2 Anti Entropy

Anti-entropy is a kind of SI class epidemic algorithm. According to that, the three states of an epidemic environment can be applied in terms of our databases:

- Susceptible: the database does not know a certain update
- Infective: the database knows a certain update and spreads it
- Removed: the database knows a certain update but does not spread it

Regularly each of our n replicas $D_1 \dots D_n$ chooses a small number m of the other $n - m$ replicas uniformly at random as synchronization partner. After choosing a site, the whole data is synchronized between these two databases. Synchronizing the whole database is expensive because of the need to transfer somehow the whole database through the network, in order to be able to

compare the data. Thus this technique can not be used too frequently. It is suggested to be an update strategy besides other techniques like the direct mail strategy.

We now present three synchronization strategies. For these, we consider S to be the database that chooses T for the anti-entropy synchronization process.

Synchronization via Push

The process of push synchronization is depicted in figure 9. If data in S is newer than the data in T, the data in T is set to the data in S. It is obvious, that the number of infective databases grows exponentially until about $\frac{n}{2}$ databases are infective. From this time on the factor of distributing the update goes down up to $1 - \frac{1}{e}$, due to the fact that there are fewer susceptible databases round by round [DGH+87].

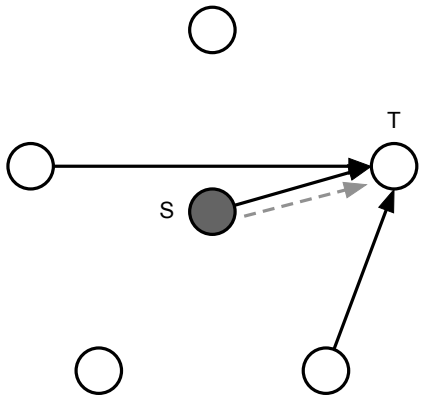

Figure 9: Synchronization via push between S and T. S has the update and infects T.

In general, the database T keeps susceptible after round i, if it was susceptible in round $i-1$ and no infective database chooses T as sync partner in round $i-1$. The probability $p_{chosen}(i)$, that a database chooses T as a sync partner in round i is:

$$p_{chosen}(i) = \left(1 - \frac{1}{n}\right)^{n \cdot (1 - p_{susceptible}(i-1))}$$

So you can build the following recursive formula for the probability $p(i)$ of T to be susceptible in round i:

$$\begin{aligned} p(i) &= p(i-1) \cdot p_{chosen}(i-1) \\ &= p(i-1) \cdot \left(1 - \frac{1}{n}\right)^{n \cdot (1 - p(i-1))} \end{aligned}$$

Based on that formula and the theoretically provable fact, that an anti-entropy will spread information to all possible recipients, in [Pit87] they have build a formula for how long it will take, to spread specific information to a large n of possible recipients by starting with only one infective update:

$$log_2(n) + ln(n) + O(1)$$

Synchronization via Pull

The pull synchronization works as shown in figure 10. If data in S is older than the data in T the data in S is set to the data in T. Now, in the first rounds, it is unpredictable how many databases

get infected, but with high probability $\frac{n}{2}$ databases will be infected in logarithmic time. After this point the chance to get infected raises to $1 - \varepsilon$.

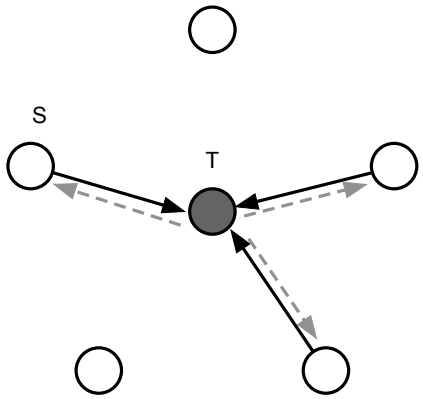

Figure 10: Synchronization via pull between S and T. T has the update and infects S.

This means, if a database remains susceptible after i rounds if it was susceptible after $i - 1$ rounds and contacted another susceptible database in round i. Thus the probability $p(i)$ to be susceptible in round i is

$$p(i) = p(i-1)^2$$

One can show that spreading the information to all databases within a network can be done in time

$$\log n + \mathcal{O}(\log \log n)$$

Synchronization via Push&Pull

In the push&pull synchronization scheme (figure 11) two databases get infected regardless of which of them holds the newer data. This means, if data in S is newer, S infects T and vice versa.

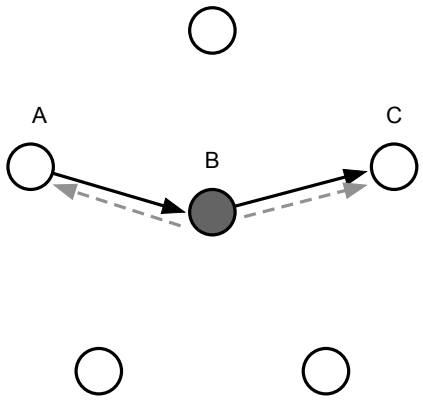

Figure 11: Synchronization via push&pull between A, B and C. B has the update and infects A and C.

When originating an update it takes at most c rounds to double the number of infected databases. After c rounds the number of infected databases grows exponentially, because in every

round each database D_i contacts another database D_{s_1} and is contacted by another database D_{s_2}, with D_{s_1} and D_{s_2} susceptible with high probability. If half of all databases are infected the number of infected databases grows at least as fast as in the pull synchronization scheme, but faster due to the combination of both schemes.

As shown in [KSSV00] the time needed to infect all databases is

$$\log_3 n + \mathcal{O}(\log \log n)$$

4.3 Rumor mongering

Rumor mongering is a SIR class epidemic algorithm, i.e. an infective database will loose the interest in spreading a certain update if some of the contacted sites have already seen this update.

Every database stores an *infective list*, containing current updates. If a database is infective – meaning that its infective list is not empty – it sends updates from this list to another site, chosen at random. The receiver tries to insert the updates into its database and adds the updates to its own infective list on success. After sending an update to some sites that have already seen this update, the sending site removes this update from its infective list. The interesting thing is, to determine when to remove an update from the infective list.

For determining when to drop updates, there are different variations of the rumor mongering algorithm:

Blind. The infective site decides to drop a update only according to the internal state of the site.

Feedback. A contacted site replies whether it knows the update already, or not. According to this reply, the infective site makes a decision.

Coin. The infective site will stop sending an update with probability $1/k$, where k is the number of contacted sites.

Counter. The sender increases a counter for every sent update and removes the update when the counter exceeds a specified value.

In case of using the Blind and Counter method there are differences between the infection schemes the algorithm can use [Sch02].

Push For the push scheme a choice of $k \geq \log_2 n + c \ln n, c > 1$ would ensure an increasing predictability for informing all sites in the network. For a reasonable choice of c the probability of informing all sites would be $1 - n^{-\mathcal{O}(1)}$, while sending $\mathcal{O}(n \log n)$ messages.

Pull With regard to the exponential growth at the start a choice of $k \in c \log_n, c > 1$ is necessary to informing all sites within $\log n + \mathcal{O}(\log \log n)$ time, while sending $\mathcal{O}(n \log n)$ messages.

Push&Pull The choice of $k = \log_3 n + n \log \log n$ guarantees a nearly complete spreading of information using the push&pull scheme, while only sending $\mathcal{O}(n \log \log n)$ messages as proved in [KSSV00].

5 Routing

In a network, messages are sent between hosts to exchange information. Due to the fact, that the source host S and the destination host D are not always in direct communication range there has to be a mechanism to deliver this messages using intermediate hosts. This process of finding a path to the communication partner is called *routing*.

Nowadays, there exist many protocols for routing in wired and wireless networks, but all of them are limited to fully connected networks. For delivering messages, these protocols (e.g. Dynamic

Source Routing (DSR) [JM96], Destination-Sequenced Distance-Vector Routing (DSDV) [PB94]) all require connected paths from S to D.

For example, considering a *mobile sensor network* where many sensors with wireless connectivity are deployed over an area. This sensors may be designed to detect chemicals, temperature, or what ever. The collected data has to be sent to a base station using intermediate nodes as routers. But due to the mobility of the sensors and limitations in communication range there is not necessarily a connected path to the base station.

The presence of such scenarios where an connection between sender an receiver is not given at every time, calls for algorithms that are able to work in such situations.

5.1 Epidemic Routing

According to these requirements and limitation, Vahdat and Becker [VB00] invented a routing protocol that makes use of an epidemic algorithm to solve the problems caused by such scenarios. The design goal was to provide message delivery with high probability and minimizing resource consumption under the given circumstances.

Description of the algorithm

```
whenever two hosts come into communication range
    if host has the lower ID
        start anti-entropy session and exchange all messages,
        that one of the hosts has not seen yet
```

Figure 12: The Epidemic Routing algorithm (raw description).

When a host S wants to send a message to another host D, S passes the message to its neighbor hosts and they do the same until the message reaches D. If there is a partition of the network on the way to D, it is likely that one (or more) host(s) of the current section come into contact with another fraction at a later time due to node movement. So, the message is passed throughout the network and will eventual received by D with high probability (figure 13).

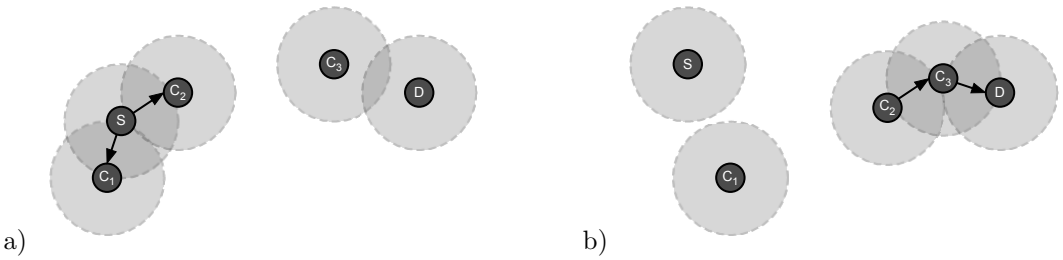

Figure 13: Message delivery from node S to D.

The exchange of information between two nodes, when they come close to each other, uses the technique of anti-entropy presented by Demers et al. in the context of updates in replicated databases [DGH+87]. Originally, anti-entropy means that a host chooses randomly another host for data exchange. For the use in Epidemic Routing, happenstance is "simulated" by the event of two hosts coming into communication range.

Each host stores messages in a buffer (hash table), indexed by the unique ID of each message. If an anti-entropy session is established, the two nodes exchange messages they do not know. For the gain of performance, only a summary vector (bit vector) is exchanged, indicating which entries in the buffer are set.

If the message buffer overflows, old messages are flushed. This has to be taken in account especially if limitations are made to the buffer size or the maximum hop count. In this case, there has to be a fair queueing strategy to achieve fairness and quality of service.

Additions

Redundant communication can be avoided by adding a host cache at every node. Therein recently seen hosts can be stored and communication to such host can be omitted.

Another addition is the *Probabilistic Routing* proposed by Lindgren et al. [LDS03], where they added the possibility to only route to hosts that are likely to meet the destination node. This is done by establishing a *delivery predictability* to every known host.

Evaluation

Vahdat and Becker implemented the Epidemic Routing protocol using the Monarch Wireless and Mobility Extensions [Mon99] to ns-2 simulator [ns204]. They simulated 50 nodes moving randomly within a 1500 m × 300 m area and examined the behaviour of their protocol with respect of various settings in transmission range (250 m, 100 m, 50 m, 25 m, 10 m), message hop limit (8, 4, 3, 2, 1 hop(s)) and message buffer size (2000, 1000, 500, 200, 100, 50, 20, 10 messages).

The results showed that given pairwise connectivity and enough buffer size (2000 equals infinite size, because the whole experiment used only 1980 messages) and time, an epidemic algorithm can guarantee eventual delivery of 100 % of messages (figure 14). One exception has to be mentioned: the experiment with a transmission range of only 10 m did not deliver all messages until the end of the experiment (within 200,000 seconds). This is due to the fact, that it is hard for the hosts to meet each other because the covered area of one host is rather small (cp. column 'coverage floor' in table 1). Given more time, 100 % of messages would be delivered according to the random movement.

They also showed, that using restrictions on the different parameters, messages are still delivered with high probability increasing delivery latency just by a little factor. For instance, limiting the hop count to three hops still ensures delivery of all messages with the average latency increasing by 33 % (figure 15). When limiting the buffer size to 2.5 % of the originated messages, the number of delivered messages decreases to 79.9 % (figure 16). But using a buffer for 5–25 % of messages was sufficient to deliver almost all messages with reasonable latency.

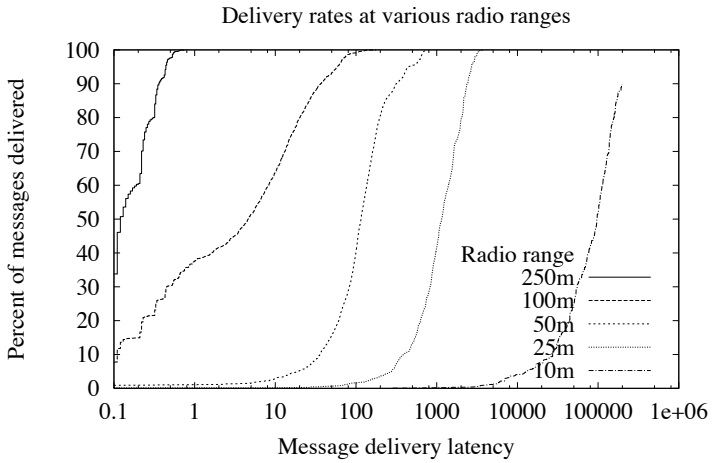

Figure 14: Message delivery with respect to transmission range [VB00].

range	delivery rate (%)	avg. latency (s)	avg. hops	coverage floor
250 m	100.0 %	0.2	2.4	10.91 %
100 m	100.0 %	12.8	6.3	1.75 %
50 m	100.0 %	153.0	3.7	0.44 %
25 m	100.0 %	618.9	3.3	0.11 %
10 m	89.9 %	44829.7	3.4	0.02 %

Table 1: Characteristics of Epidemic Routing [VB00].

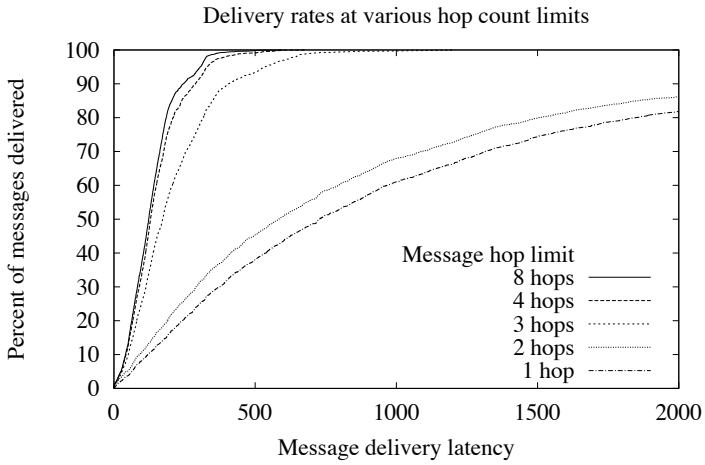

Figure 15: Message delivery with respect to message hop limit at 50 m transmission range [VB00].

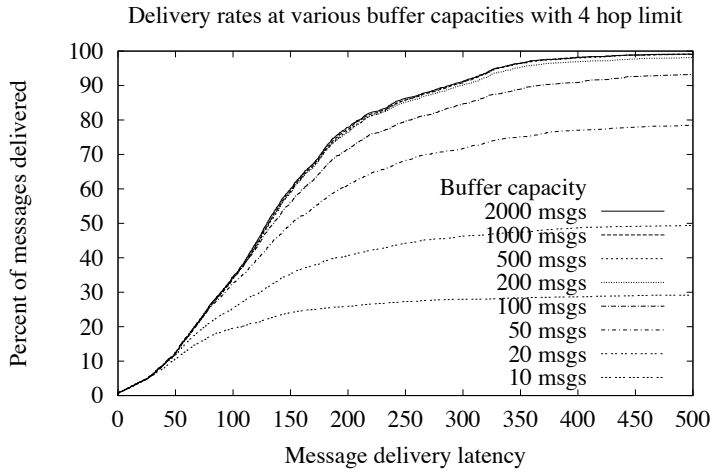

Figure 16: Message delivery with respect to message buffer size [VB00].

buffer size	delivery rate (%)	avg. latency (s)
2000	100.0	147.3
1000	100.0	148.7
500	100.0	149.2
200	99.6	152.0
100	95.2	157.5
50	79.7	148.2
20	50.2	129.5
10	29.3	98.9

Table 2: Delivery rate and latency with 50 m transmission range and a limit of 4 hops at various buffer sizes [VB00].

Classification

Epidemic Routing can be classified as a SIS class epidemic algorithm. Every host is *susceptible* for a specific message M_i and gets *infected* by this message. If the buffer size of the host is small, it will flush old messages due to buffer overflow and will be *susceptible* for that message M_i again.

Given infinite storage space (which is the same as having enough buffer size for all messages originated in a specific environment) a host would be infective forever and Epidemic Routing would be in the class of SI algorithms. If the routing protocol implements a mechanism for acknowledgements as mentioned by Vahdat and Becker [VB00] it would be SIR, because hosts would be informed about successful message delivery and become recovered.

References

[DGH+87] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing*, pages 1–12, 1987.

[JM96] David B. Johnson and David A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*, pages 153–181. Kluwer Academic Publishers, 1996.

[Kep94] Jeffrey O. Kephart. A biologically inspired immune system for computers. In *roceedings of the Fourth International Workshop on Synthesis and Simulatoin of Living Systems*, pages 130–139, August 1994.

[KSSV00] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading, 2000.

[LDS03] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):19–20, 2003.

[Mon99] The Rice University Monarch Project. Rice Monarch mobility and wireless extensions, 1999. http://www.monarch.cs.rice.edu/cmu-ns.html.

[ns204] The Network Simulator – ns-2, 2004. http://www.isi.edu/nsnam/ns/.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of the SIGCOM '94 Conference on Communications Architecture, Protocols and Applications*, pages 234–244, August 1994.

[Pit87] Boris Pittel. On spreading a rumor. *SIAM J. Appl. Math.*, 47(1):213–223, 1987.

[Sch02] Christian Schindelhauer. Algorithmische grundlagen des internets. Lecture notes, University of Paderborn, 2002.

[VB00] Amin Vahdat and David Becker. Epidemic routing for partially connected ad hoc networks. Technical report CS-200006, Duke University, April 2000. http://issg.cs.duke.edu/epidemic/.